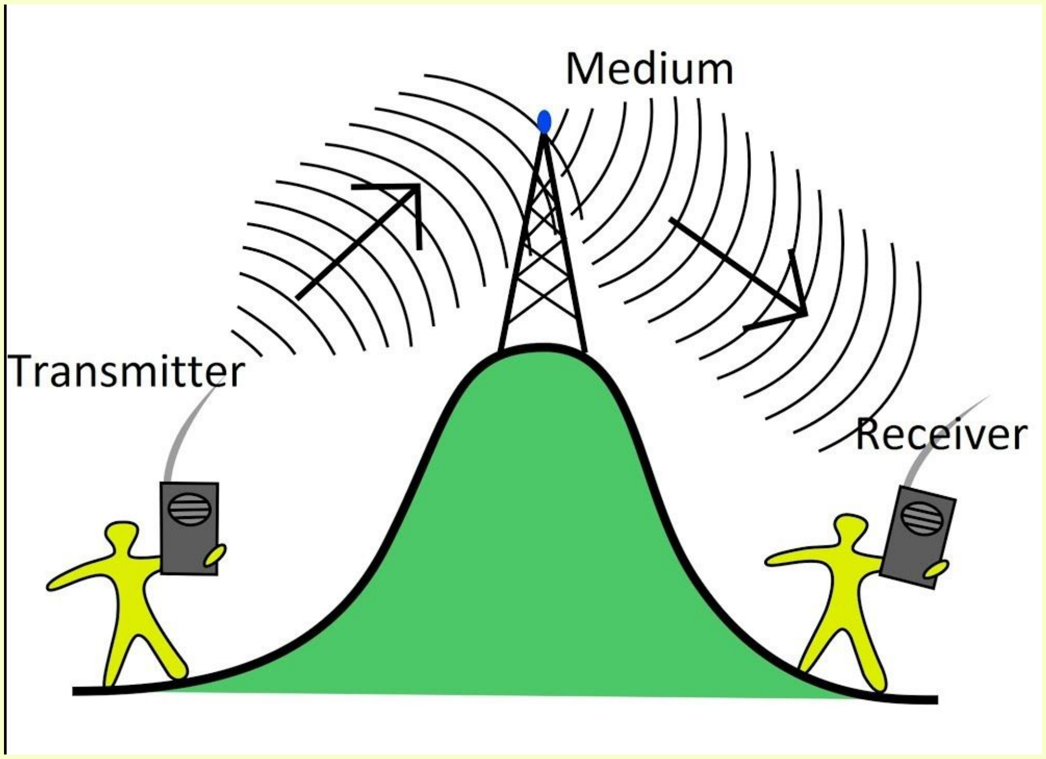
Zewail City of Science and Technology
University of Science and Technology
CIE 428 - fall 2019

Assignment 3



BY:

Alaa Hesham Mahmoud	201500638

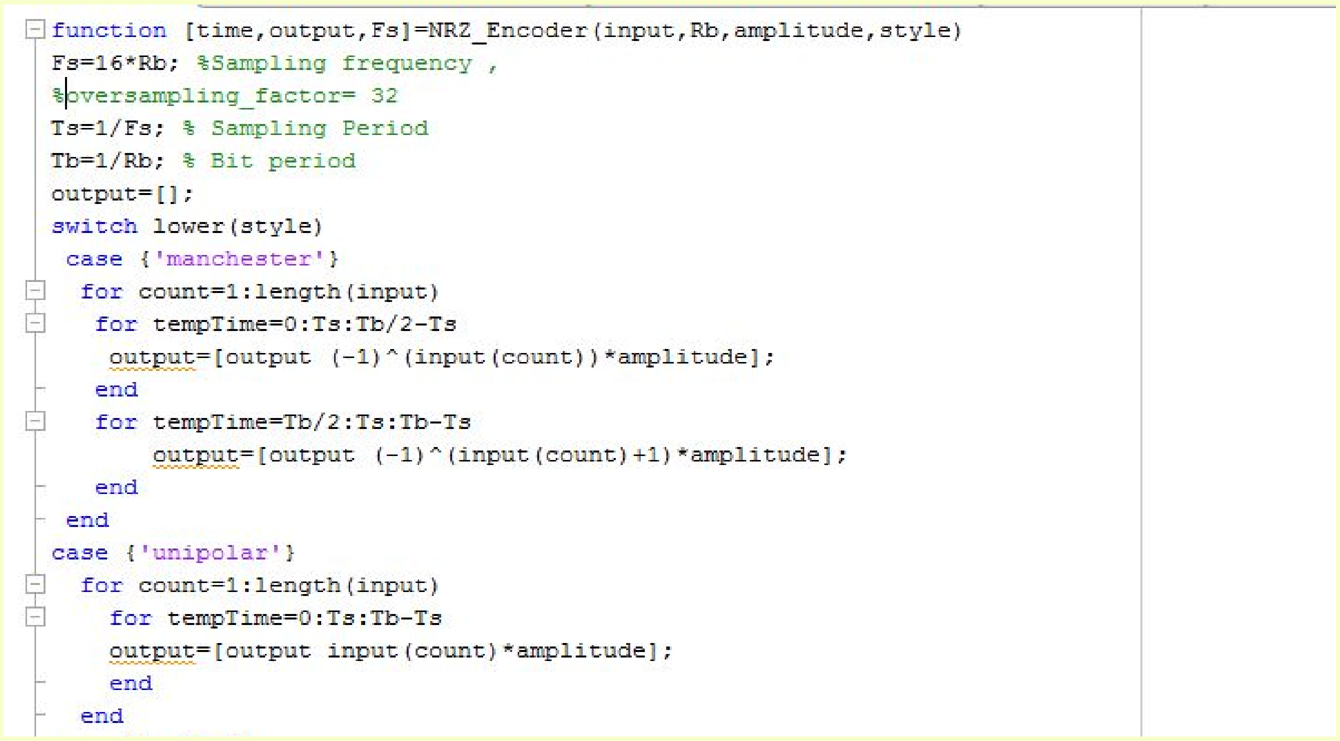
Saturday,November 23, 2019

1 | BPSK CODE

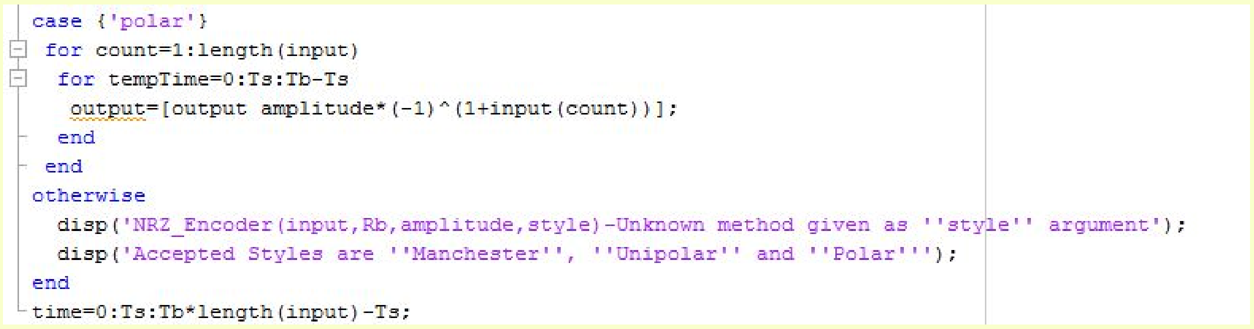
Overview

Code highlights :

- Generate data using Randn Matlab function (1s and 0s)
- Perform data Encoding using polar NRZ scheme using NRZ_Encoder(1 ⟶ symbol 1 ,-1 ⟶ symbol 0)
- Bpsk modulation is equivalent to multiplying data by sinusoidal carrier. If symbol 1 is transmitted, it will be cos(2*pi*fc*t)/sin(2*pi*fc*t) . if symbol 0 is transmitted , it will be -cos (2*pi*fc*t)
- To simulated channel effect , AWGN is summed to Bpsk modulated signals
- For Bpsk receiver , multiply received bits with carrier elementwise , and then integrate with the aid of Matlab Built -in function Trapz.
- To calculate psd , matlab built-in"psd" function is used.
- To calculate the bit error rate(BER) , Xoring data with streamed/received bits and divide it by the total data size to get percentage of error.
- To plot SNR against BER , we will assume that *E_b/N0 in dB* vary from -6 to 10 and calculate corresponding BER using this formula to get theoretical *BER = 0.5*erfc(sqrt(10.^(EbN0dB/10)))*

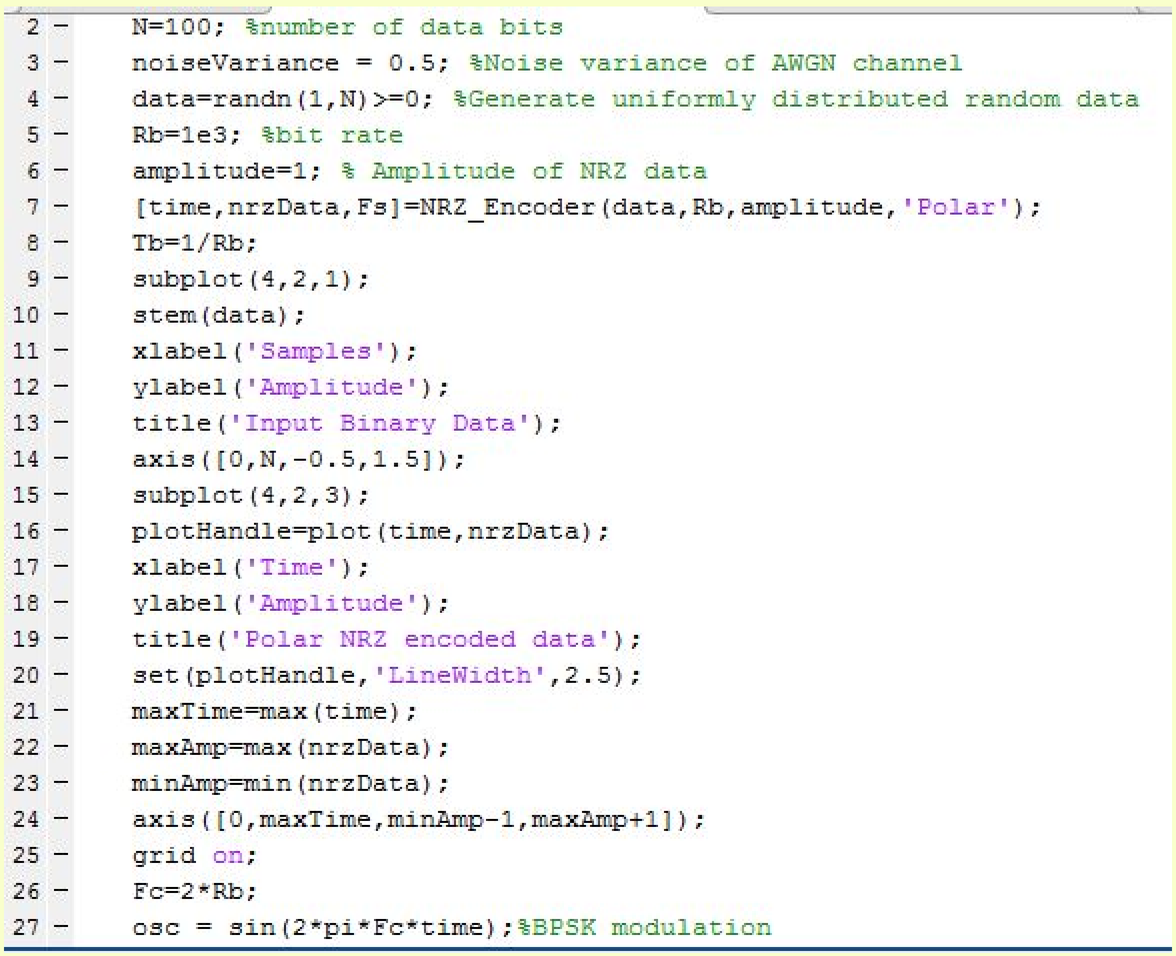


```
function [time,output,Fs]=NRZ_Encoder(input,Rb,amplitude,style)
Fs=16*Rb; %Sampling frequency ,
%oversampling_factor= 32
Ts=1/Fs; % Sampling Period
Tb=1/Rb; % Bit period
output=[];
switch lower(style)
 case {'manchester'}
  for count=1:length(input)
   for tempTime=0:Ts:Tb/2-Ts
    output=[output (-1)^(input(count))*amplitude];
   end
   for tempTime=Tb/2:Ts:Tb-Ts
       output=[output (-1)^(input(count)+1)*amplitude];
   end
 end
case {'unipolar'}
  for count=1:length(input)
    for tempTime=0:Ts:Tb-Ts
    output=[output input(count)*amplitude];
    end
  end
```

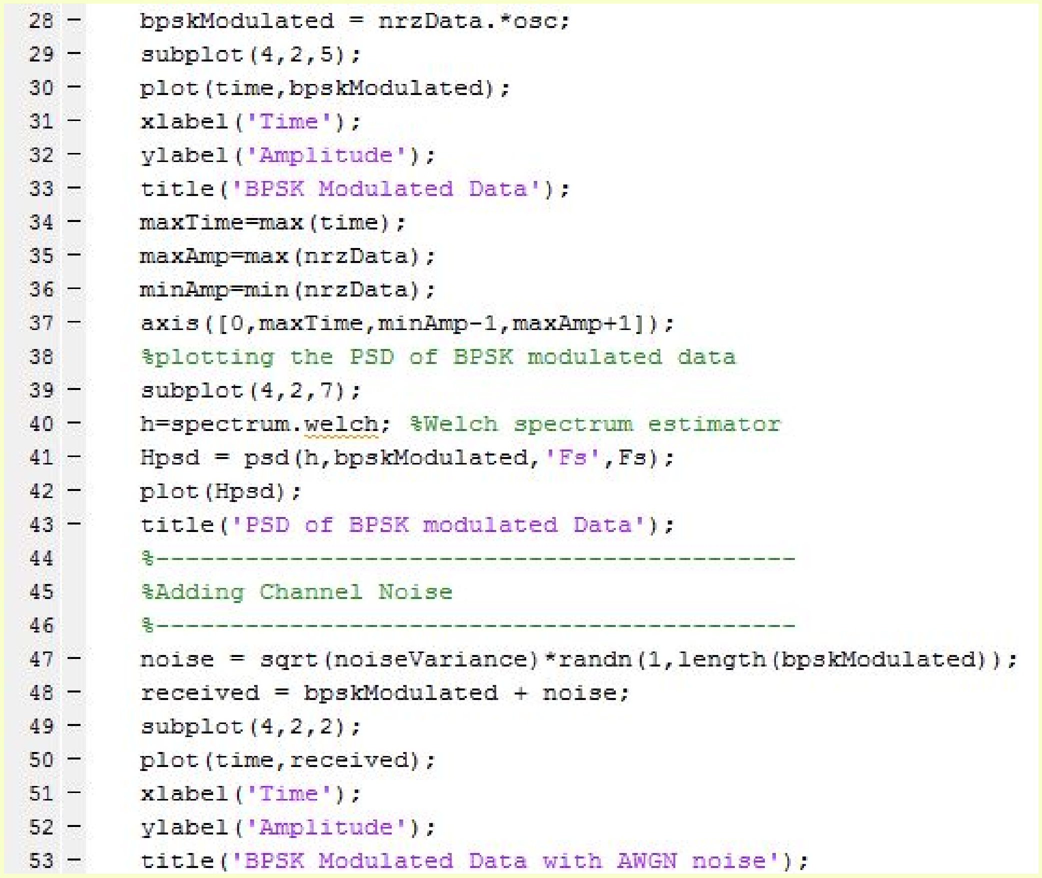


```
case {'polar'}
 for count=1:length(input)
  for tempTime=0:Ts:Tb-Ts
   output=[output amplitude*(-1)^(1+input(count))];
  end
 end
otherwise
  disp('NRZ_Encoder(input,Rb,amplitude,style)-Unknown method given as ''style'' argument');
  disp('Accepted Styles are ''Manchester'', ''Unipolar'' and ''Polar''');
end
time=0:Ts:Tb*length(input)-Ts;
```

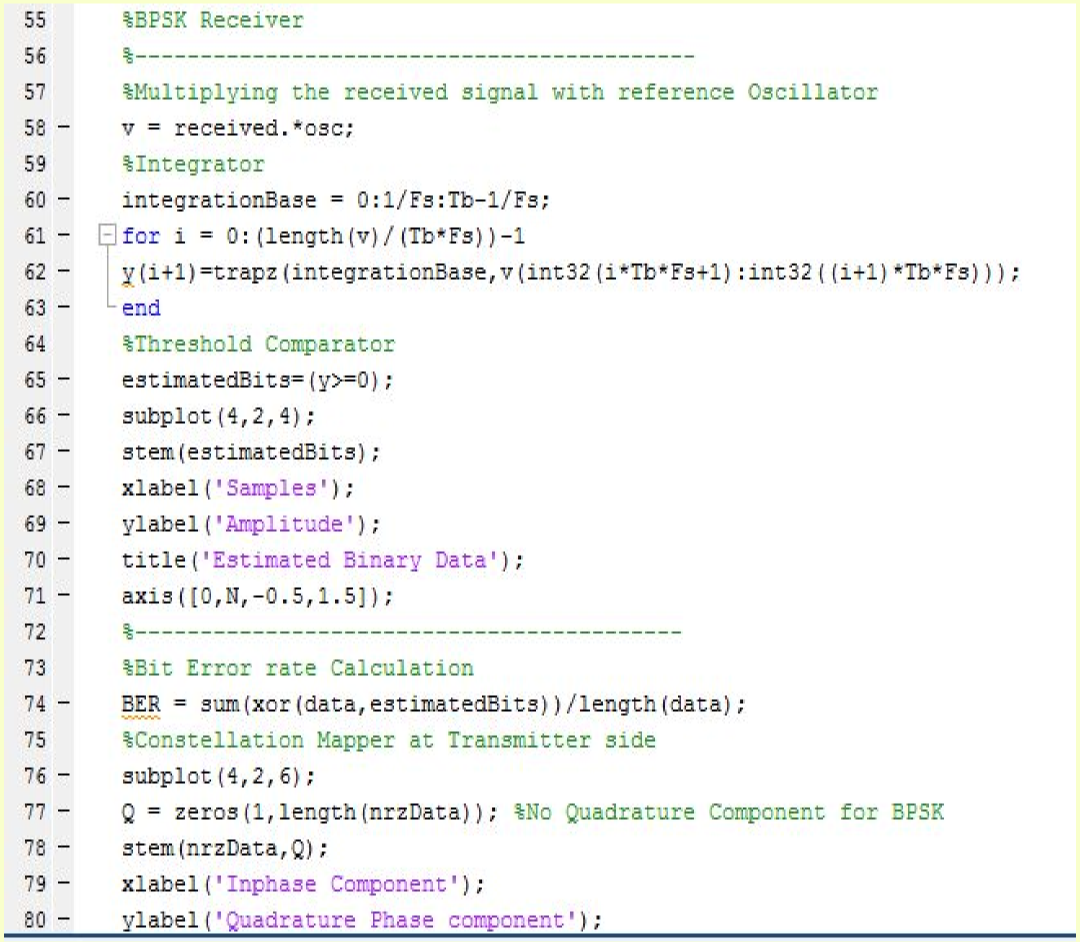
Encoder NRZ



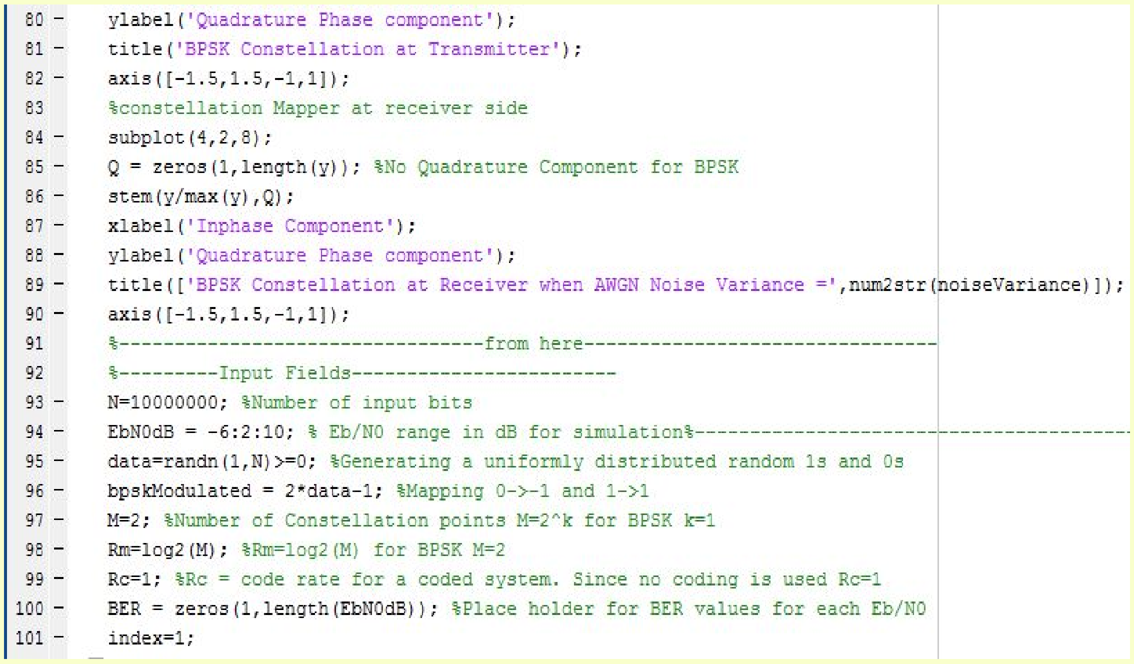
```
N=100; %number of data bits
noiseVariance = 0.5; %Noise variance of AWGN channel
data=randn(1,N)>=0; %Generate uniformly distributed random data
Rb=1e3; %bit rate
amplitude=1; % Amplitude of NRZ data
[time,nrzData,Fs]=NRZ_Encoder(data,Rb,amplitude,'Polar');
Tb=1/Rb;
subplot(4,2,1);
stem(data);
xlabel('Samples');
ylabel('Amplitude');
title('Input Binary Data');
axis([0,N,-0.5,1.5]);
subplot(4,2,3);
plotHandle=plot(time,nrzData);
xlabel('Time');
ylabel('Amplitude');
title('Polar NRZ encoded data');
set(plotHandle,'LineWidth',2.5);
maxTime=max(time);
maxAmp=max(nrzData);
minAmp=min(nrzData);
axis([0,maxTime,minAmp-1,maxAmp+1]);
grid on;
Fc=2*Rb;
osc = sin(2*pi*Fc*time);%BPSK modulation
```



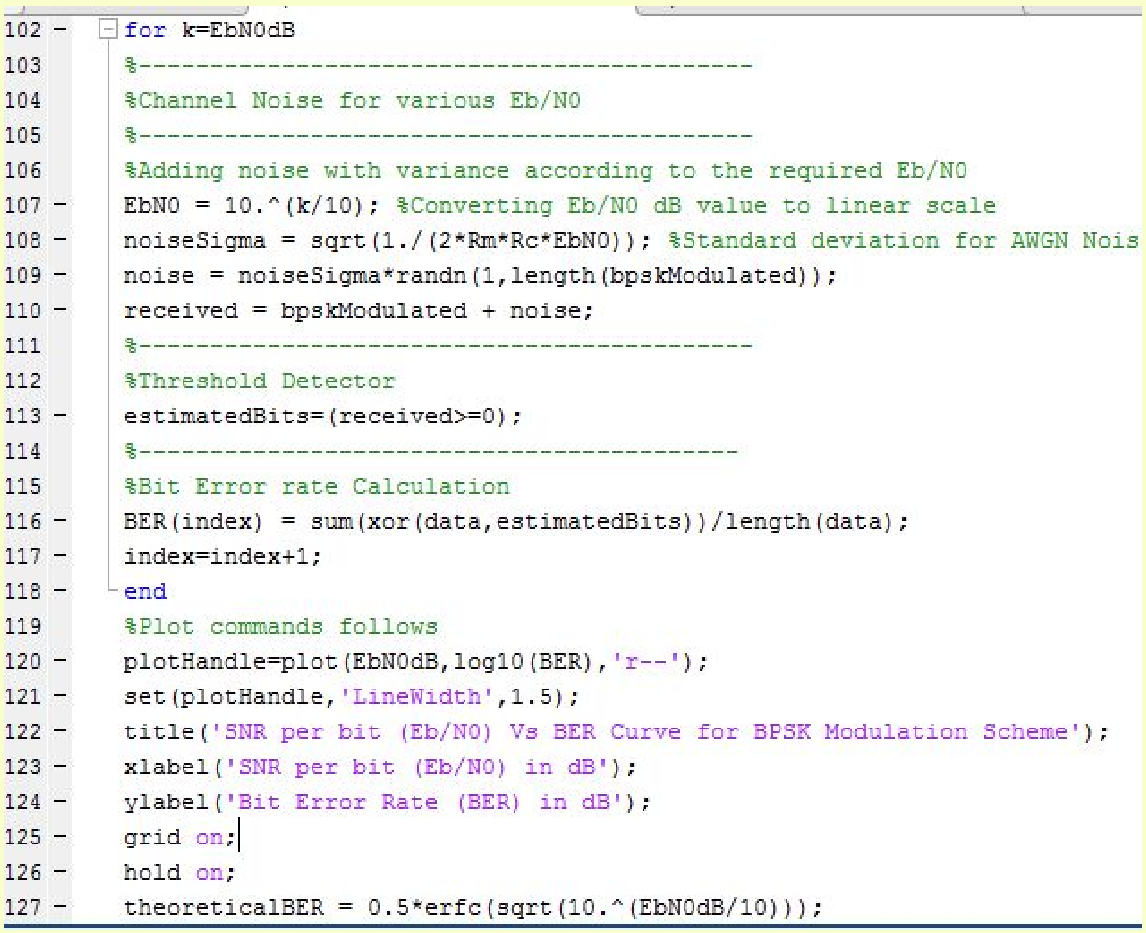
```matlab
bpskModulated = nrzData.*osc;
subplot(4,2,5);
plot(time,bpskModulated);
xlabel('Time');
ylabel('Amplitude');
title('BPSK Modulated Data');
maxTime=max(time);
maxAmp=max(nrzData);
minAmp=min(nrzData);
axis([0,maxTime,minAmp-1,maxAmp+1]);
%plotting the PSD of BPSK modulated data
subplot(4,2,7);
h=spectrum.welch; %Welch spectrum estimator
Hpsd = psd(h,bpskModulated,'Fs',Fs);
plot(Hpsd);
title('PSD of BPSK modulated Data');
%-------------------------------------------
%Adding Channel Noise
%-------------------------------------------
noise = sqrt(noiseVariance)*randn(1,length(bpskModulated));
received = bpskModulated + noise;
subplot(4,2,2);
plot(time,received);
xlabel('Time');
ylabel('Amplitude');
title('BPSK Modulated Data with AWGN noise');
```



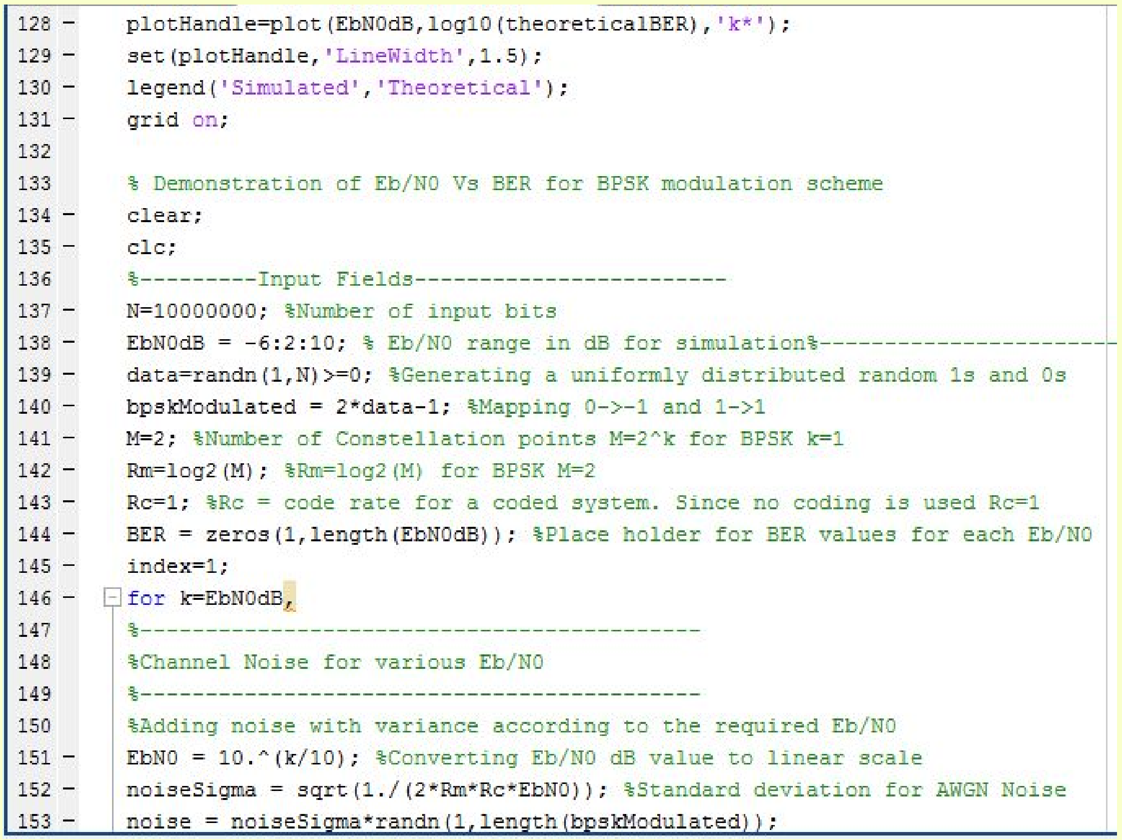
```
%BPSK Receiver
%------------------------------------------
%Multiplying the received signal with reference Oscillator
v = received.*osc;
%Integrator
integrationBase = 0:1/Fs:Tb-1/Fs;
for i = 0:(length(v)/(Tb*Fs))-1
y(i+1)=trapz(integrationBase,v(int32(i*Tb*Fs+1):int32((i+1)*Tb*Fs)));
end
%Threshold Comparator
estimatedBits=(y>=0);
subplot(4,2,4);
stem(estimatedBits);
xlabel('Samples');
ylabel('Amplitude');
title('Estimated Binary Data');
axis([0,N,-0.5,1.5]);
%------------------------------------------
%Bit Error rate Calculation
BER = sum(xor(data,estimatedBits))/length(data);
%Constellation Mapper at Transmitter side
subplot(4,2,6);
Q = zeros(1,length(nrzData)); %No Quadrature Component for BPSK
stem(nrzData,Q);
xlabel('Inphase Component');
ylabel('Quadrature Phase component');
```



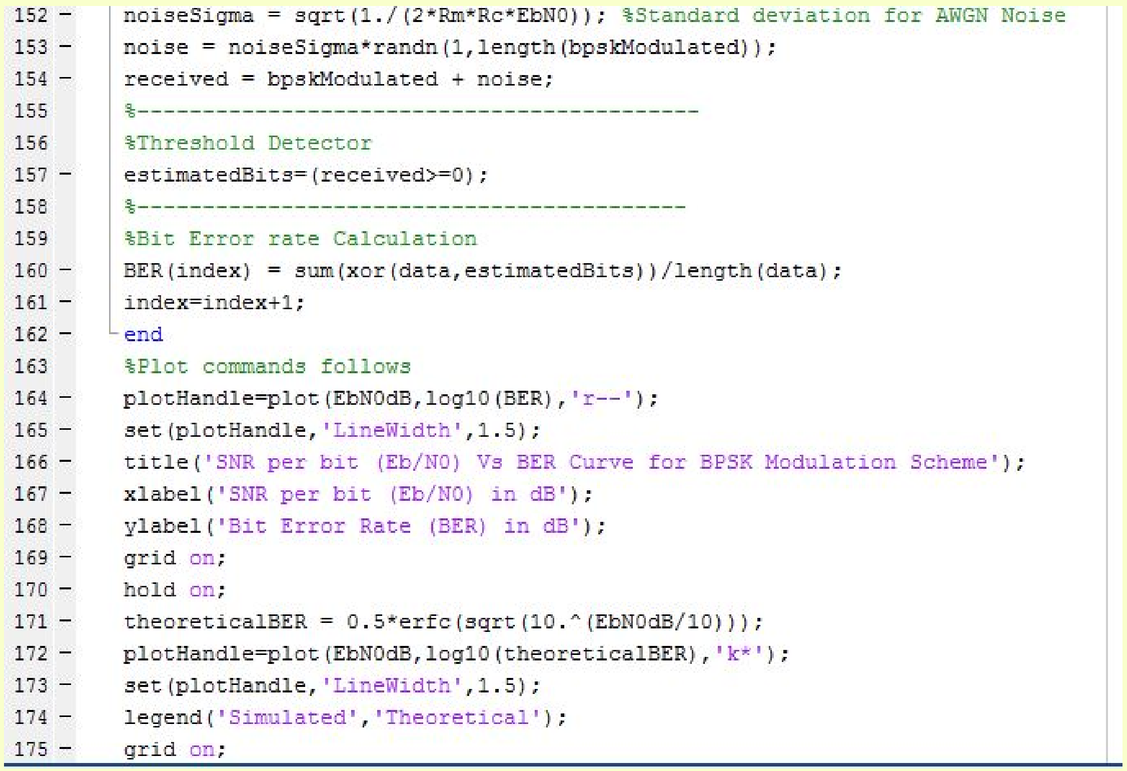
```
ylabel('Quadrature Phase component');
title('BPSK Constellation at Transmitter');
axis([-1.5,1.5,-1,1]);
%constellation Mapper at receiver side
subplot(4,2,8);
Q = zeros(1,length(y)); %No Quadrature Component for BPSK
stem(y/max(y),Q);
xlabel('Inphase Component');
ylabel('Quadrature Phase component');
title(['BPSK Constellation at Receiver when AWGN Noise Variance =',num2str(noiseVariance)]);
axis([-1.5,1.5,-1,1]);
%--------------------------------from here--------------------------------
%---------Input Fields------------------------
N=10000000; %Number of input bits
EbN0dB = -6:2:10; % Eb/N0 range in dB for simulation%-----------------------------------------
data=randn(1,N)>=0; %Generating a uniformly distributed random 1s and 0s
bpskModulated = 2*data-1; %Mapping 0->-1 and 1->1
M=2; %Number of Constellation points M=2^k for BPSK k=1
Rm=log2(M); %Rm=log2(M) for BPSK M=2
Rc=1; %Rc = code rate for a coded system. Since no coding is used Rc=1
BER = zeros(1,length(EbN0dB)); %Place holder for BER values for each Eb/N0
index=1;
```



```
for k=EbN0dB
    %--------------------------------------------
    %Channel Noise for various Eb/N0
    %--------------------------------------------
    %Adding noise with variance according to the required Eb/N0
    EbN0 = 10.^(k/10); %Converting Eb/N0 dB value to linear scale
    noiseSigma = sqrt(1./(2*Rm*Rc*EbN0)); %Standard deviation for AWGN Nois
    noise = noiseSigma*randn(1,length(bpskModulated));
    received = bpskModulated + noise;
    %--------------------------------------------
    %Threshold Detector
    estimatedBits=(received>=0);
    %-------------------------------------------
    %Bit Error rate Calculation
    BER(index) = sum(xor(data,estimatedBits))/length(data);
    index=index+1;
end
%Plot commands follows
plotHandle=plot(EbN0dB,log10(BER),'r--');
set(plotHandle,'LineWidth',1.5);
title('SNR per bit (Eb/N0) Vs BER Curve for BPSK Modulation Scheme');
xlabel('SNR per bit (Eb/N0) in dB');
ylabel('Bit Error Rate (BER) in dB');
grid on;
hold on;
theoreticalBER = 0.5*erfc(sqrt(10.^(EbN0dB/10)));
```



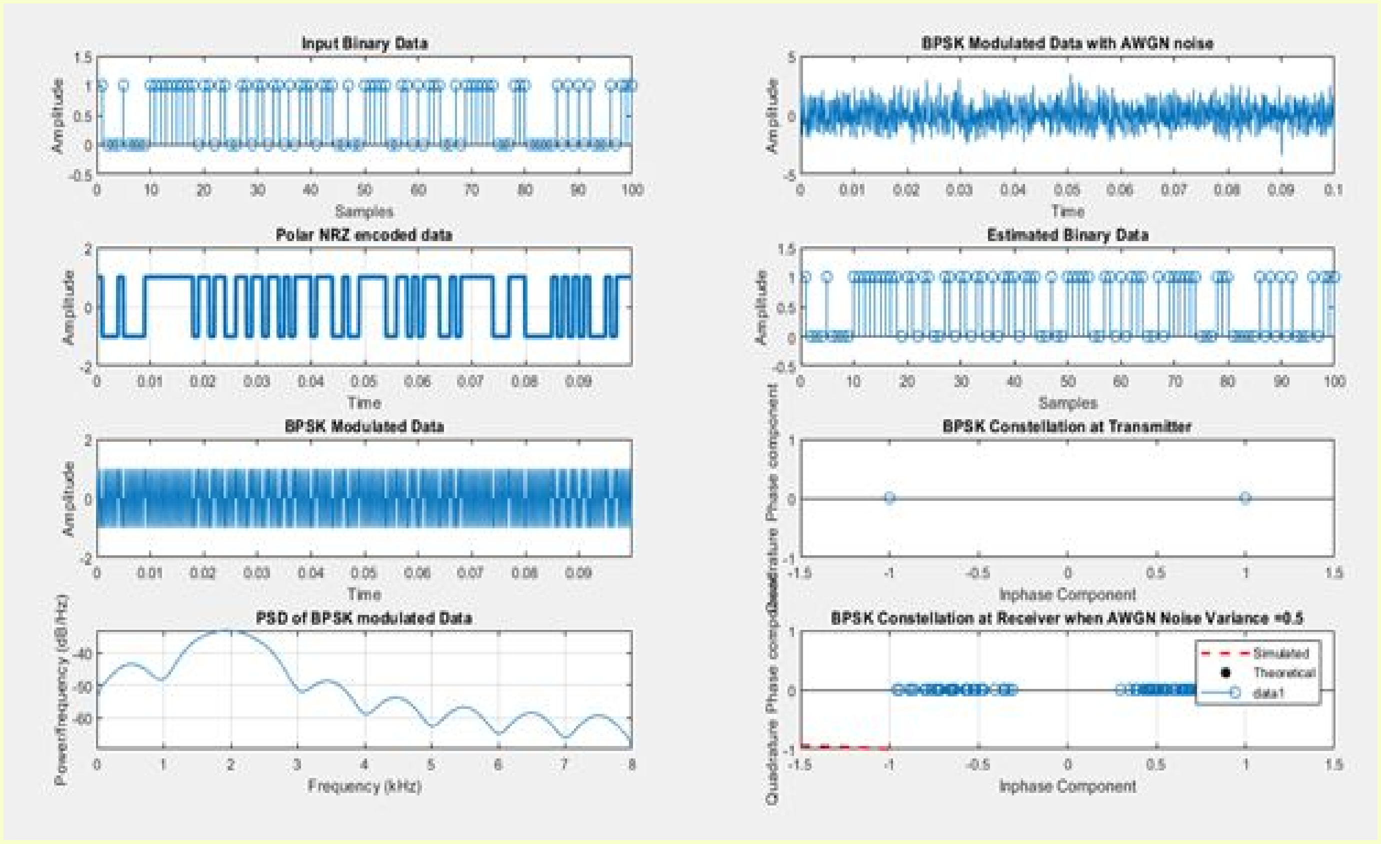
```
plotHandle=plot(EbN0dB,log10(theoreticalBER),'k*');
set(plotHandle,'LineWidth',1.5);
legend('Simulated','Theoretical');
grid on;

% Demonstration of Eb/N0 Vs BER for BPSK modulation scheme
clear;
clc;
%---------Input Fields------------------------
N=10000000; %Number of input bits
EbN0dB = -6:2:10; % Eb/N0 range in dB for simulation%----------------------
data=randn(1,N)>=0; %Generating a uniformly distributed random 1s and 0s
bpskModulated = 2*data-1; %Mapping 0->-1 and 1->1
M=2; %Number of Constellation points M=2^k for BPSK k=1
Rm=log2(M); %Rm=log2(M) for BPSK M=2
Rc=1; %Rc = code rate for a coded system. Since no coding is used Rc=1
BER = zeros(1,length(EbN0dB)); %Place holder for BER values for each Eb/N0
index=1;
for k=EbN0dB,
%-------------------------------------------
%Channel Noise for various Eb/N0
%-------------------------------------------
%Adding noise with variance according to the required Eb/N0
EbN0 = 10.^(k/10); %Converting Eb/N0 dB value to linear scale
noiseSigma = sqrt(1./(2*Rm*Rc*EbN0)); %Standard deviation for AWGN Noise
noise = noiseSigma*randn(1,length(bpskModulated));
```



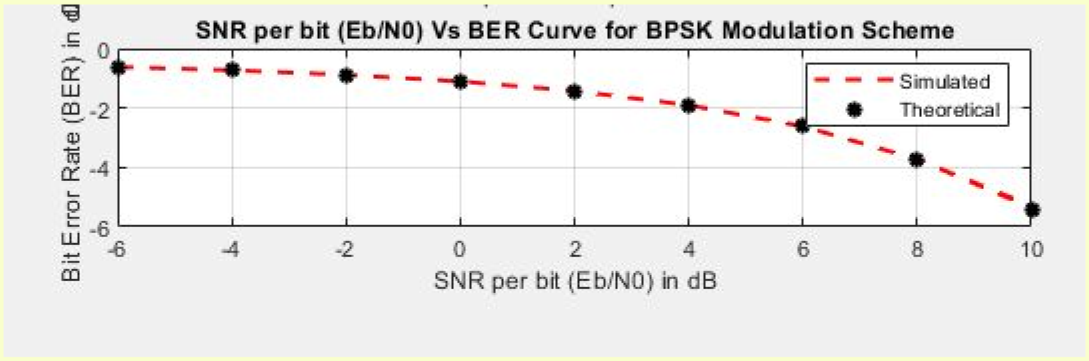
noiseSigma = sqrt(1./(2*Rm*Rc*EbN0)); %Standard deviation for AWGN Noise
noise = noiseSigma*randn(1,length(bpskModulated));

```
received = bpskModulated + noise;
%-------------------------------------------
%Threshold Detector
estimatedBits=(received>=0);
%-------------------------------------------
%Bit Error rate Calculation
BER(index) = sum(xor(data,estimatedBits))/length(data);
index=index+1;
end
%Plot commands follows
plotHandle=plot(EbN0dB,log10(BER),'r--');
set(plotHandle,'LineWidth',1.5);
title('SNR per bit (Eb/N0) Vs BER Curve for BPSK Modulation Scheme');
xlabel('SNR per bit (Eb/N0) in dB');
ylabel('Bit Error Rate (BER) in dB');
grid on;
hold on;
theoreticalBER = 0.5*erfc(sqrt(10.^(EbN0dB/10)));
plotHandle=plot(EbN0dB,log10(theoreticalBER),'k*');
set(plotHandle,'LineWidth',1.5);
legend('Simulated','Theoretical');
grid on;
```

2 | BPSK Figures



All figures required are plotted in this subplot except SNR



SNR per bit in dB versus BER in dB

3| Bpsk results comment

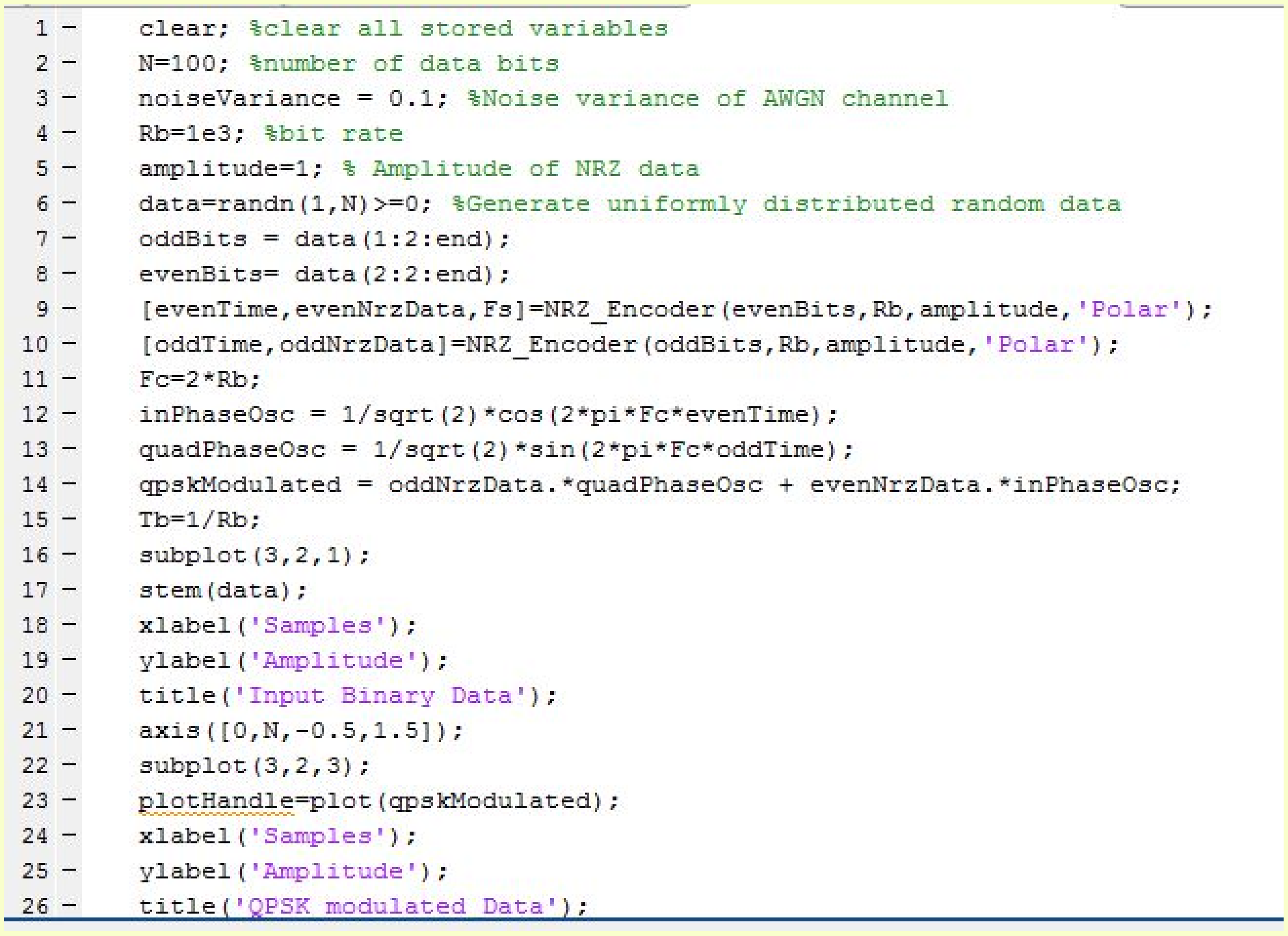
I think all the results make sense :

- for input binary data , they are either 0s or 1s which is the case .
- For polar NRZ encoded data , they have the same shape as binary data except that for symbol 0 they have a negative amplitude instead of zero .
- For BPSK , they have multiplied by a carrier that is why they have a sinusoidal shape , we could easily detect phase discontinuity .
- For PSD , most power is concentrated at 2 kH which is apparently carrier frequency . There are side loops as pulse shape in time domain is a rect function so it will be sinc in frequency domain.
- After adding noise with variance equal to 0.5 , we could easily notice that data amplitude is not constant anymore . Since we are doing phase modulation , it will not affect Estimated Binary data that much .
- For constellation , it is as expected have one basis function , and symbol one corresponds to(1* E_b*basis function) and symbol 0 corresponds to (-1*E_b*basis function) .Here we have made E_b equals to one .
- For constellation after adding noise , we can see that noise makes data deviate from E_b and -E_b that is why we use maximum likelihood to decide if a noisy symbol is symbol 0or symbol 1 .
- For the SNR graph versus BER , it makes sense .The more you add power to signal , the distance between symbols increase and hence the probability of error decreases .

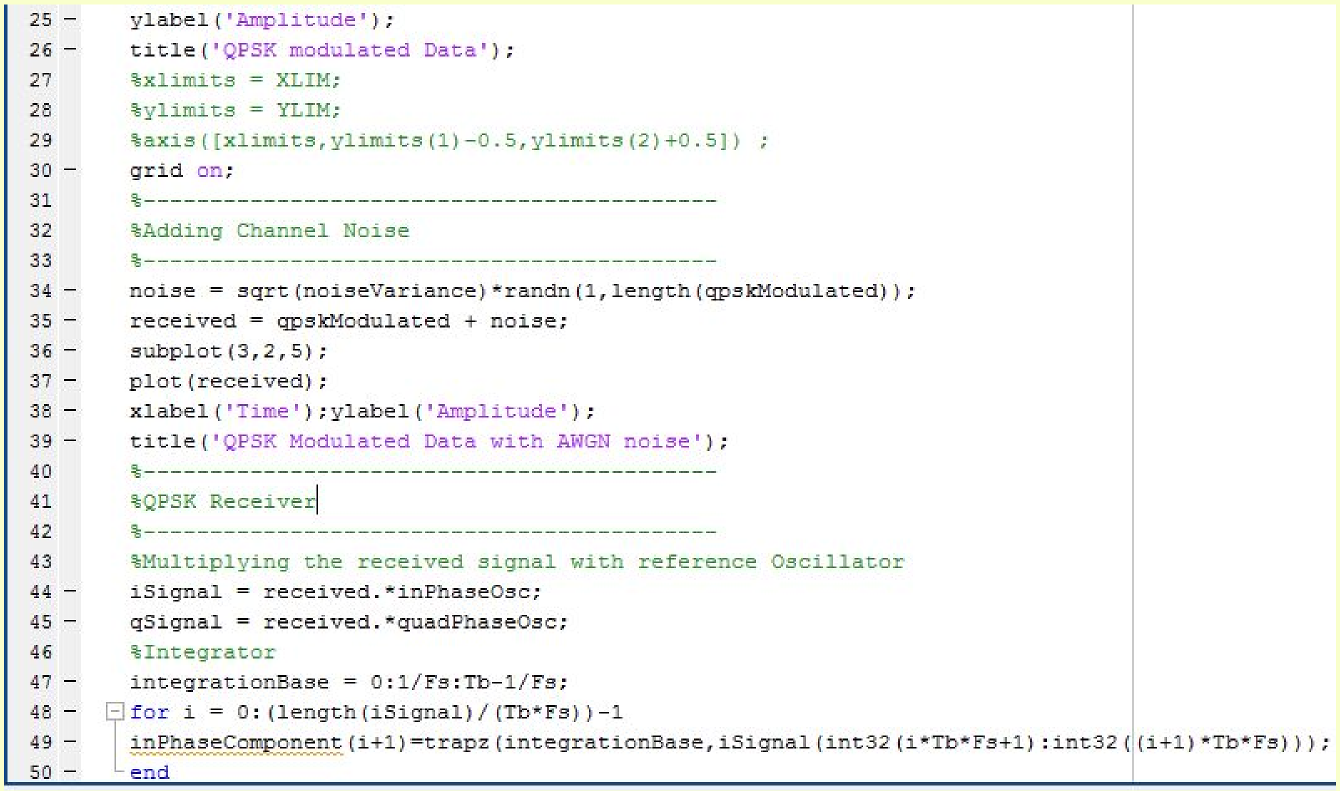
4 | Qpsk Code

Code highlights :

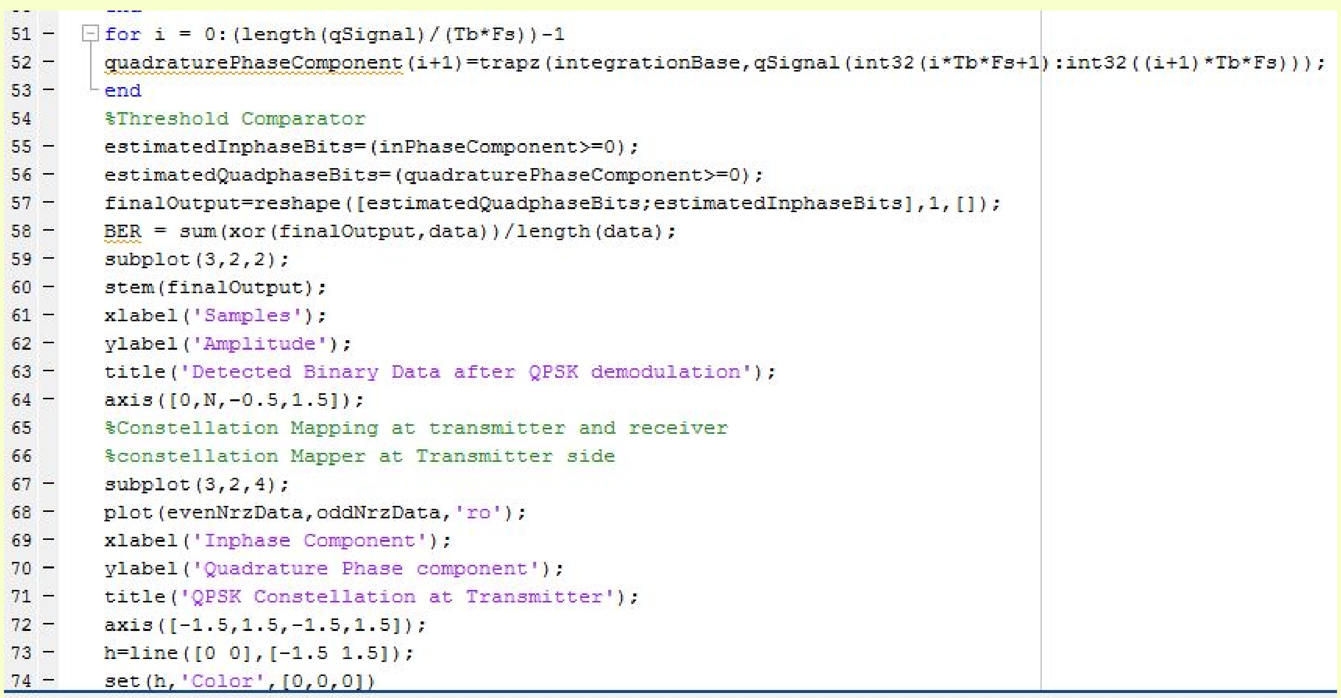
- Generate data using Randn Matlab function (1s and 0s).
- Divide bits to even bits and odd bits as well as Time so that inphase components are 1/sqrt(2)*cos(2*pi*Fc*evenTime), and quadrature components are 1/sqrt(2)*sin(2*pi*Fc*oddTime) .
- QPSK modulated signals is the summation of in-phase and quadrature components .
- To simulated channel effect , AWGN is summed to Qpsk modulated signals.However this noise is now has two components real component to be added to inphase components and imaginary components to be added its magnitude to quadrature components .
- For Qpsk receiver , multiply received bits with carrier elementwise .For in phase components multiply them by in phase oscillator and for quadrature multiply them by quadrature oscillator , and then integrate them separately with the aid of Matlab Built -in function Trapz .
- To calculate psd , matlab built-in"psd" function is used.
- To calculate the bit error rate(BER) , Xoring data with streamed/received bits and divide it by the total data size to get percentage of error.



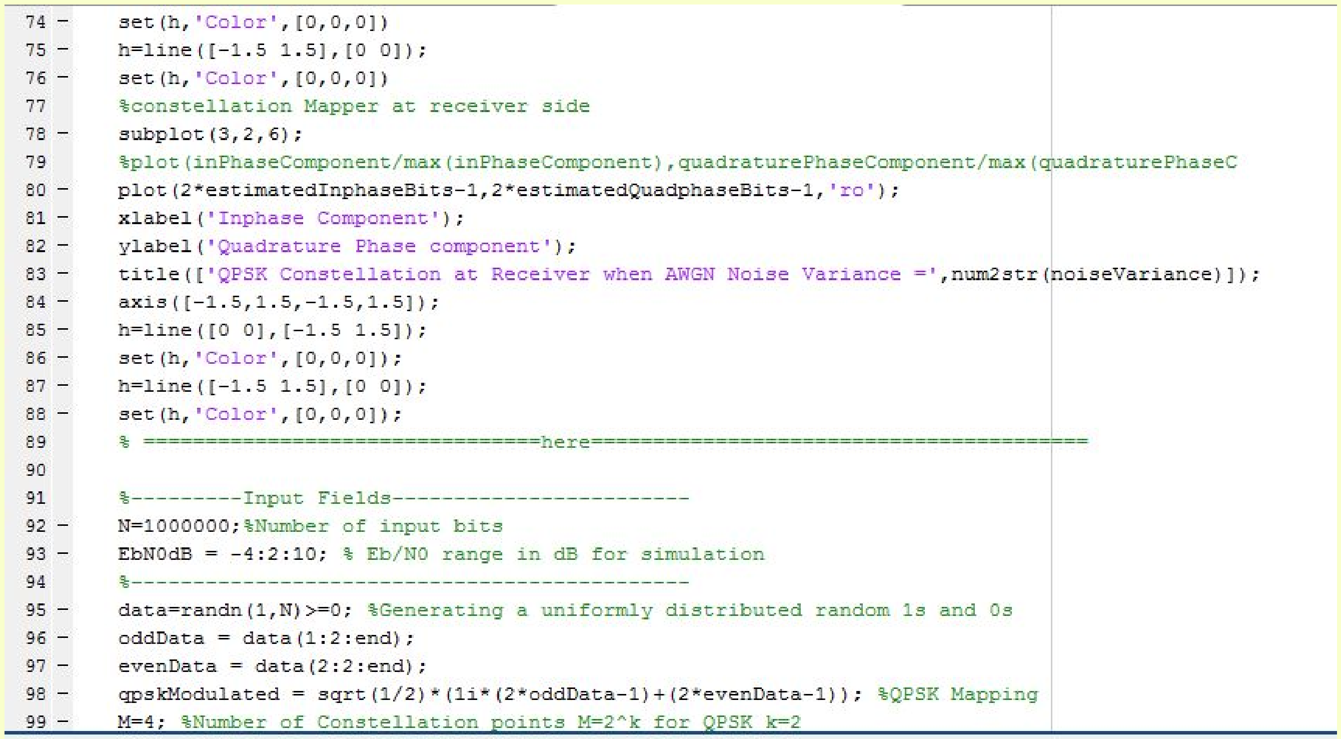
```
clear; %clear all stored variables
N=100; %number of data bits
noiseVariance = 0.1; %Noise variance of AWGN channel
Rb=1e3; %bit rate
amplitude=1; % Amplitude of NRZ data
data=randn(1,N)>=0; %Generate uniformly distributed random data
oddBits = data(1:2:end);
evenBits= data(2:2:end);
[evenTime,evenNrzData,Fs]=NRZ_Encoder(evenBits,Rb,amplitude,'Polar');
[oddTime,oddNrzData]=NRZ_Encoder(oddBits,Rb,amplitude,'Polar');
Fc=2*Rb;
inPhaseOsc = 1/sqrt(2)*cos(2*pi*Fc*evenTime);
quadPhaseOsc = 1/sqrt(2)*sin(2*pi*Fc*oddTime);
qpskModulated = oddNrzData.*quadPhaseOsc + evenNrzData.*inPhaseOsc;
Tb=1/Rb;
subplot(3,2,1);
stem(data);
xlabel('Samples');
ylabel('Amplitude');
title('Input Binary Data');
axis([0,N,-0.5,1.5]);
subplot(3,2,3);
plotHandle=plot(qpskModulated);
xlabel('Samples');
ylabel('Amplitude');
title('QPSK modulated Data');
```



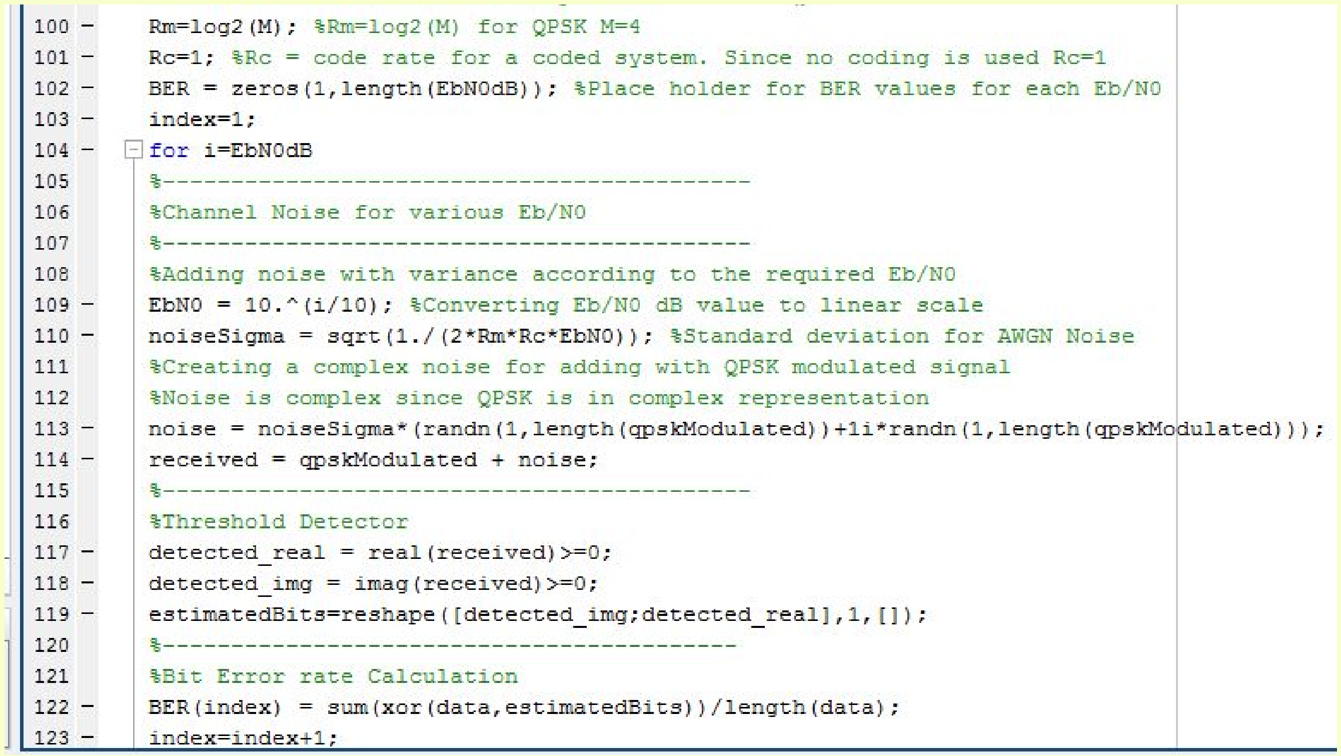
```
ylabel('Amplitude');
title('QPSK modulated Data');
%xlimits = XLIM;
%ylimits = YLIM;
%axis([xlimits,ylimits(1)-0.5,ylimits(2)+0.5]) ;
grid on;
%-------------------------------------------
%Adding Channel Noise
%-------------------------------------------
noise = sqrt(noiseVariance)*randn(1,length(qpskModulated));
received = qpskModulated + noise;
subplot(3,2,5);
plot(received);
xlabel('Time');ylabel('Amplitude');
title('QPSK Modulated Data with AWGN noise');
%-------------------------------------------
%QPSK Receiver
%-------------------------------------------
%Multiplying the received signal with reference Oscillator
iSignal = received.*inPhaseOsc;
qSignal = received.*quadPhaseOsc;
%Integrator
integrationBase = 0:1/Fs:Tb-1/Fs;
for i = 0:(length(iSignal)/(Tb*Fs))-1
inPhaseComponent(i+1)=trapz(integrationBase,iSignal(int32(i*Tb*Fs+1):int32((i+1)*Tb*Fs)));
end
```



```
for i = 0:(length(qSignal)/(Tb*Fs))-1
quadraturePhaseComponent(i+1)=trapz(integrationBase,qSignal(int32(i*Tb*Fs+1):int32((i+1)*Tb*Fs)));
end
%Threshold Comparator
estimatedInphaseBits=(inPhaseComponent>=0);
estimatedQuadphaseBits=(quadraturePhaseComponent>=0);
finalOutput=reshape([estimatedQuadphaseBits;estimatedInphaseBits],1,[]);
BER = sum(xor(finalOutput,data))/length(data);
subplot(3,2,2);
stem(finalOutput);
xlabel('Samples');
ylabel('Amplitude');
title('Detected Binary Data after QPSK demodulation');
axis([0,N,-0.5,1.5]);
%Constellation Mapping at transmitter and receiver
%constellation Mapper at Transmitter side
subplot(3,2,4);
plot(evenNrzData,oddNrzData,'ro');
xlabel('Inphase Component');
ylabel('Quadrature Phase component');
title('QPSK Constellation at Transmitter');
axis([-1.5,1.5,-1.5,1.5]);
h=line([0 0],[-1.5 1.5]);
set(h,'Color',[0,0,0])
```

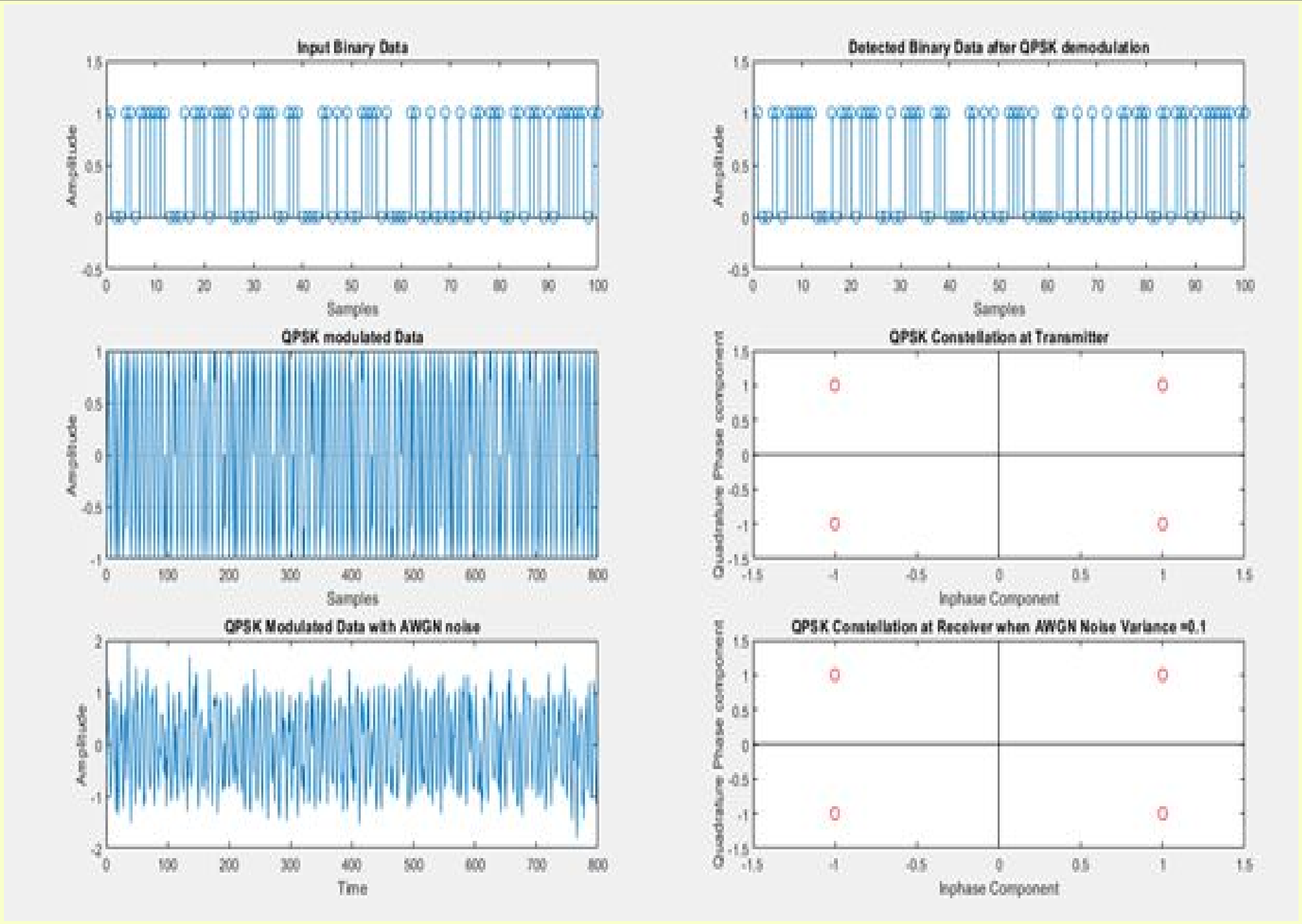


```
set(h,'Color',[0,0,0])
h=line([-1.5 1.5],[0 0]);
set(h,'Color',[0,0,0])
%constellation Mapper at receiver side
subplot(3,2,6);
%plot(inPhaseComponent/max(inPhaseComponent),quadraturePhaseComponent/max(quadraturePhaseC
plot(2*estimatedInphaseBits-1,2*estimatedQuadphaseBits-1,'ro');
xlabel('Inphase Component');
ylabel('Quadrature Phase component');
title(['QPSK Constellation at Receiver when AWGN Noise Variance =',num2str(noiseVariance)]);
axis([-1.5,1.5,-1.5,1.5]);
h=line([0 0],[-1.5 1.5]);
set(h,'Color',[0,0,0]);
h=line([-1.5 1.5],[0 0]);
set(h,'Color',[0,0,0]);
% ================================here========================================

%---------Input Fields------------------------
N=1000000;%Number of input bits
EbN0dB = -4:2:10; % Eb/N0 range in dB for simulation
%---------------------------------------------
data=randn(1,N)>=0; %Generating a uniformly distributed random 1s and 0s
oddData = data(1:2:end);
evenData = data(2:2:end);
qpskModulated = sqrt(1/2)*(1i*(2*oddData-1)+(2*evenData-1)); %QPSK Mapping
M=4; %Number of Constellation points M=2^k for QPSK k=2
```

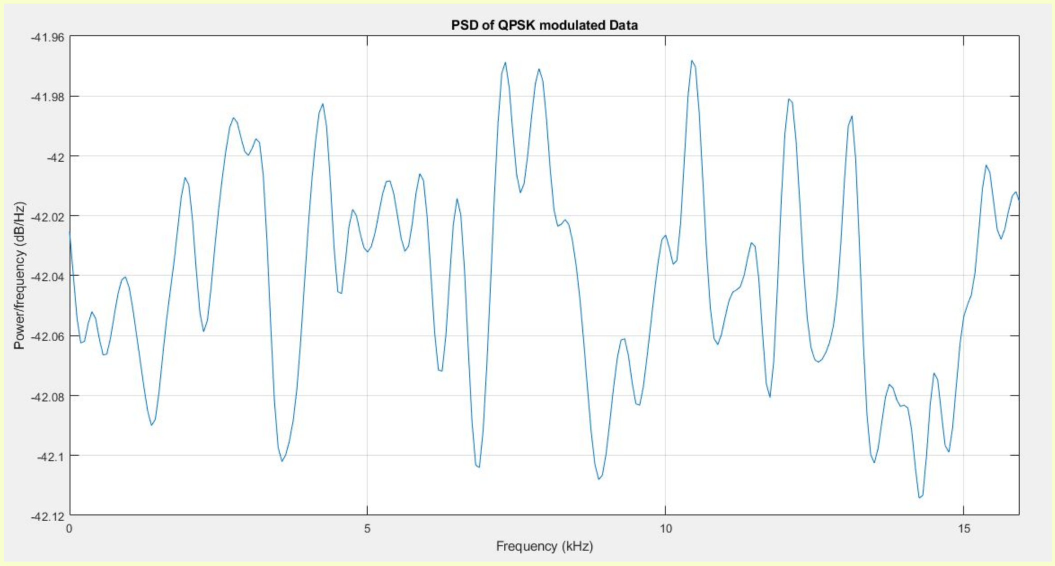


```
Rm=log2(M); %Rm=log2(M) for QPSK M=4
Rc=1; %Rc = code rate for a coded system. Since no coding is used Rc=1
BER = zeros(1,length(EbN0dB)); %Place holder for BER values for each Eb/N0
index=1;
for i=EbN0dB
%-------------------------------------------
%Channel Noise for various Eb/N0
%-------------------------------------------
%Adding noise with variance according to the required Eb/N0
EbN0 = 10.^(i/10); %Converting Eb/N0 dB value to linear scale
noiseSigma = sqrt(1./(2*Rm*Rc*EbN0)); %Standard deviation for AWGN Noise
%Creating a complex noise for adding with QPSK modulated signal
%Noise is complex since QPSK is in complex representation
noise = noiseSigma*(randn(1,length(qpskModulated))+1i*randn(1,length(qpskModulated)));
received = qpskModulated + noise;
%-------------------------------------------
%Threshold Detector
detected_real = real(received)>=0;
detected_img = imag(received)>=0;
estimatedBits=reshape([detected_img;detected_real],1,[]);
%------------------------------------------
%Bit Error rate Calculation
BER(index) = sum(xor(data,estimatedBits))/length(data);
index=index+1;
```

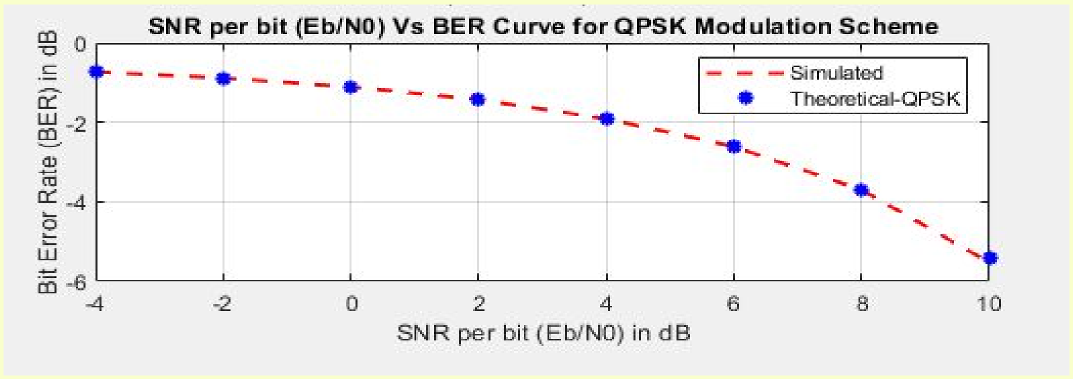
5 | Qpsk Figures



All figures required in Q-psk are plotted in this subplot   except SNR



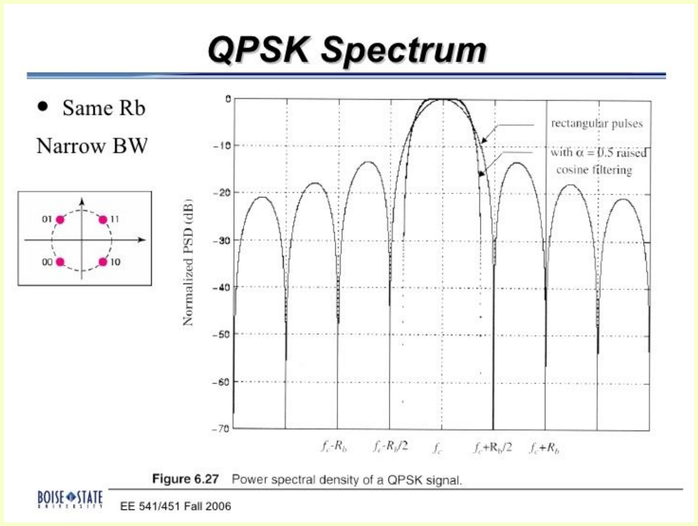
PSD of QPSK modulated Data



SNR per bit in db versus BER in db

6 | Qpsk Results comment

- for input binary data , they are either 0s or 1s which is the case .
- For QPSK , they have multiplied by a carrier that is why they have a sinusoidal shape , we could easily detect phase discontinuity .
- For PSD , I do not think that I have coded right as I was expecting to have two peaks at f+fc , and f-fc so it should be sth like this .



- After adding noise with variance equal to 0.1 , we could easily notice that data amplitude is not constant anymore . Since we are doing phase modulation , it will not affect Estimated Binary data that much .
- For constellation , it is as expected have two basis functions and the angle between every symbol is 90 . They are at (E_b,E_b),(-Eb,Eb),(-E_b,-Eb),(Eb,-Eb) where E_b =1
- For the SNR graph versus BER , it makes sense .The more you add power to signal , the distance between symbols increase and hence the probability of error decreases .

7 | References

[1]Viswanathan, M. (2013). Simulation of digital communication systems using matlab. 2nd ed. Mathuranathan Viswanathan.